DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114 was filed in this application after a decision by the Board of Patent Appeals and Interferences, but before the filing of a Notice of Appeal to the Court of Appeals for the Federal Circuit or the commencement of a civil action. Since this application is eligible for continued examination under 37 CFR 1.114 and the fee set forth in 37 CFR 1.17(e) has been timely paid, the appeal has been withdrawn pursuant to 37 CFR 1.114 and prosecution in this application has been reopened pursuant to 37 CFR 1.114. Applicant's submission filed on 5/14/2010 has been entered.

2. Claims 1, 5, 6, 9, 11, 15, 17, 19-21 and 23-34 are pending and have been examined.

EXAMINER'S AMENDMENT

3. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Malgorzata Kulczycka, Reg. No. 50,496, on June 14, 2010. During the interview, it was agreed that amendments to the independent claims to more fully reflect the invention as depicted in Fig. 2 would provide allowable subject matter.

The application has been amended as follows:

IN THE CLAIMS

Please amend the independent claims 1, 5, 15, and 19 as follows:

1. (Currently Amended) A computer-implemented method of dynamically generating web pages, said method comprising:

analyzing a page that includes static markup text and a set of code instructions executable on a server;

extracting the static markup text from the page and storing the static markup text in a resource file;

generating a servlet class for the page based on the set of code instructions, wherein the servlet class comprises a static initializer for initializing a static class array ~~variable~~ for the servlet class, and wherein the servlet class does not include the static markup text;

in response to a first use of the servlet class by any instance of the servlet class:

invoking the static class initializer of the servlet class, wherein the invoking of the static class initializer of the servlet class causes the static markup text to be read from the resource file, and the static class array ~~variable~~ to be initialized with the static markup text;

hotloading ~~loading~~ a copy of the servlet class comprising the static class array ~~variable~~ initialized with the static markup text into shared memory;

in response to each request of a plurality of requests for the page from a plurality of clients, performing the steps of:

instantiating a distinct instance of the servlet class on the server, wherein instantiating each instance of the servlet class does not create another copy of the static markup text;

executing said distinct instance of the servlet class, wherein execution of each instance of the server class generates a compiled page by combining the static markup text that resides in the shared memory with results produced by executing the set of code instructions; and

sending the compiled page to a client that requested the page;

wherein the method is performed by one or more computing devices.

5. (Currently Amended) A method of initiating a first instance of an application that shares a set of static markup text with other instances of the application, wherein the first instance of the application is generated by compiling code from a page that contains both the code and the set of static markup text in response to a request from one or more users, said method comprising:

executing instructions to instantiate the first instance of the application, wherein said instructions are stored on a non-transitory computer-readable storage medium, said instructions that, when executed, cause one or more processors to perform the steps of:

analyzing the page to extract the set of static markup text and storing the set of static markup text in a resource file;

generating a servlet class for the page based on the code from the page, wherein the servlet class comprises a static initializer for initializing a ~~set of~~ static class array ~~variables~~ for the servlet class, and wherein the servlet class does not include the set of static markup text;

in response to a first use of the servlet class by any instance of the servlet class:

invoking the static class initializer of the servlet class, wherein the invoking of the static class initializer of the servlet class causes the set of static markup text to be read from the resource file, and the ~~set of~~ static class array ~~variables~~ to be initialized with the set of static markup text;

hotloading ~~loading~~ a copy of the servlet class comprising the ~~set of~~ static class array ~~variables~~ initialized with the set of static markup text into shared, read-only memory;

in response to each request of a plurality of requests for the page from the one or more users, performing the steps of:

instantiating a distinct instance of the servlet class, wherein instantiating each instance of the servlet class does not create another copy of the set of static markup text;

executing said distinct instance of the servlet class, wherein execution of each instance of the server class generates a compiled page based on the copy of by combining the set of static markup text that

resides in the shared, read-only memory with results produced by executing the set of code instructions;

sending the compiled page to a client that requested the page; and

accessing the set of static markup text in the shared, read-only memory when the code from the first instance of the application is executed;

wherein the method is performed by one or more computing devices.

15. (Currently Amended) A non-transitory computer-readable storage medium storing instructions that, when executed, cause one or more processors to perform a method for sharing static markup text from a page among a plurality of users in response to requests from said plurality of users, said method comprising:

analyzing a page that includes static markup text and a set of code instructions executable on a server;

extracting the static markup text from the page and storing the static markup text in a resource file;

generating a servlet class for the page based on the set of code instructions, wherein the servlet class comprises a static initializer for initializing a static class array ~~variable~~ for the servlet class, and wherein the servlet class does not include the static markup text;

in response to a first use of the servlet class by any instance of the servlet class:

invoking the static class initializer of the servlet class, wherein the invoking of the static class initializer of the servlet class causes the static markup text to be read from the resource file, and the static class array ~~variable~~ to be initialized with the static markup text;

hotloading ~~loading~~ a copy of the servlet class comprising the static class array ~~variable~~ initialized with the static markup text into shared memory;

in response to each request of a plurality of requests for the page from a plurality of clients, performing the steps of:

instantiating a distinct instance of the servlet class on the server, wherein instantiating each instance of the servlet class does not create another copy of the static markup text;

executing said distinct instance of the servlet class, wherein execution of each instance of the server class generates a compiled page by combining the static markup text that resides in the shared memory with results produced by executing the set of code instructions; and

sending the compiled page to a client that requested the page.

19. (Currently Amended) A non-transitory computer-readable storage medium storing instructions that, when executed, cause one or more processors to perform a method of initiating a first instance of an application that shares a set of static markup text with other instances of the application, wherein the first instance of the application is

generated by compiling code from a page that contains both the code and the set of static markup text in response to a request from one or more users, said method comprising:

executing instructions to instantiate the first instance of the application;

wherein the instructions to instantiate the first instance of the application include:

analyzing the page to extract the set of static markup text and storing the set of static markup text in a resource file;

generating a servlet class for the page based on the code from the page, wherein the servlet class comprises a static initializer for initializing a ~~set of~~ static class array ~~variables~~ for the servlet class, and wherein the servlet class does not include the static markup text;

in response to a first use of the servlet class by any instance of the servlet class:

invoking the static class initializer of the servlet class, wherein the invoking of the static class initializer of the servlet class causes the set of static markup text to be read from the resource file, and the ~~set of~~ static class array ~~variables~~ to be initialized with the set of static markup text;

hotloading ~~loading~~ a copy of the servlet class comprising the ~~set of~~ static class array ~~variables~~ initialized with the set of static markup text into shared, read-only memory;

in response to each request of a plurality of requests for the page from the one or more users, performing the steps of:

instantiating a distinct instance of the servlet class, wherein instantiating each instance of the servlet class does not create another copy of the ~~set of~~ static class array ~~variables~~ initialized with the set of static markup text;

executing said distinct instance of the servlet class, wherein execution of each instance of the server class generates a compiled page by combining the set of static markup text that resides in the shared, read-only memory with results produced by executing the set of code instructions; sending the compiled page to the user that requested the page; and

accessing the ~~set of~~ static class array ~~variables~~ initialized with the set of static markup text in the shared, read-only memory when the code from the first instance of the application is executed.

REASONS FOR ALLOWANCE

4. The following is an examiner's statement of reasons for allowance:

The examiner indicated that this application would be in condition for allowance if the independent claims 1, 5, 15, and 19 are amended to include the features of: a static initializer for initializing a static class array for the servlet class and hotloading a copy of the servlet class comprising the static class array initialized with the static markup text into shared memory. These features are generally depicted in Fig. 2 and the associated text in the originally filed disclosure. In light of Applicant's arguments on pages 11-13 filed 5/14/2010, the above features, taken in combination with all remaining features of the independent claim are not taught or suggested by the prior art of record. The applicant agreed to amend the independent claims 1, 5, 15, and 19 as indicated by the examiner. The remaining claims 6, 9, 11, 17, 20-21 and 23-34 are each dependent upon one the independent claims 1, 5, 15, and 19, and are allowable for the same reasons. Thus, all pending claims 1, 5, 6, 9, 11, 15, 17, 19-21 and 23-34 are allowed.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

5. Any inquiry concerning this communication or earlier communications from the examiner should be directed to JAMES RUTTEN whose telephone number is (571)272-3703. The examiner can normally be reached on M-F 10:00-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571)272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/J. Derek Rutten/
Primary Examiner, Art Unit 2192